1.单选题（2分）

中国的国家顶级域名是( ).

A. .cn

B. .ch

C. .chn

D. .china

2.单选题（1.5分）

下列**不属于**面向对象程序设计语言的是( ).

A. C

B. C++

C. Java

D. C#

3.单选题（2分）

以比较作为基本运算，在N个数中找出最大数，最坏情况下所需要的最少的比较次数为（ ）。

A.N^2

B.N

C.N - 1

D.N + 1

4.单选题（2分）

表达式a* (b+c)* d 的后缀表达式为（ ），其中“*”和“+”是运算符。

A.**a+bcd

B.abc+* d*

C.abc+d**

D.* a*+bcd

5.单选题（1.5分）

在 8 位二进制补码中，10101011 表示的数是十进制下的( ).

A. 43

B. -85

C. -43

D. -84

6.单选题（1.5分）

FTP 可以用于( )。

A. 远程传输文件

B. 发送电子邮件

C. 浏览网页

D. 网上聊天

7.单选题（2分）

319和377的最大公约数是( )。

A. 27

B. 33

C. 29

D. 31

8.单选题（2分）

对于有n个顶点、m条边的无向联通图（m>n），需要删掉（ ）条边才能使其成为一棵树。

A. n - 1

B. B.m - n

C. C.m - n - 1

D. D.m - n + 1

9.单选题（1.5分）

如果一棵二叉树的中序遍历是 BAC，那么它的先序遍历不可能是( )。

A. ABC

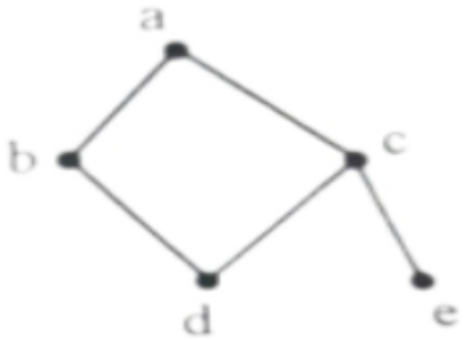
B. CBA

C. ACB

D. BAC

10.单选题（2分）

以a为起点，对右边的无向图进行深度优先遍历，则b、c、d、e四个点中有可能作为最后一个遍历到的点个数为（ ）。



A.1

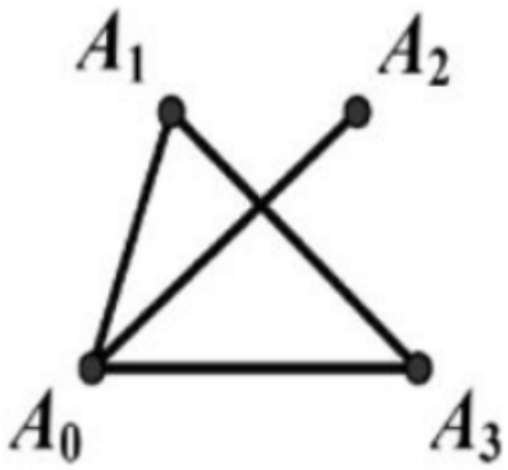
B.2

C.3

D.4

11.单选题（1.5分）

以 A0作为起点，对下面的无向图进行深度优先遍历时，遍历顺序不可能是( )。



A. A0, A1, A2, A3

B. A0, A1, A3, A2

C. A0, A2, A1, A3

D. A0, A3, A1, A2

12.单选题（2分）

6个人，两个人组一队，总共组成三只不区分队伍的编号，不同的组队情况有（）种。

A.10

B.15

C.30

D.20

13.单选题（1.5分）

如果根的高度为 1,具有 61 个结点的完全二叉树的高度为( )。

A.5

B.6

C.7

D.8

14.单选题（2分）

设 x=true, y=true, z=false，以下逻辑运算表达式值为真的是( ).

A. (y ∨ z) ∧ x ∧ z

B. B. x ∧ (z ∨ y) ∧z

C. C. (x ∧ y) ∧ z

D. D. (x ∧ y) ∨ (z ∨ x)

15.单选题（2分）

有五副不同颜色的手套（共 10 只手套，每副手套左右手各 1 只），一次性从中取 6 只手套，请问恰好能配成两副手套的不同取法有（ ）种。

A. 120

B. 180

C. 150

D. 30

16.填空题(12分)

```
#include <cstdio>
#include <cstring>
using namespace std;
char st[100];
int main() {
    scanf("%s", st);
    int n = strlen(st);
    for (int i = 1; i <= n; ++i) {
        if (n % i == 0) {
            char c = st[i - 1];
            if (c >= 'a')
                st[i - 1] = c - 'a' + 'A';
        }
    }
    printf("%s", st);
    return 0;
}
```

•判断题（1.5分一个)

1）输入的字符串只能由小写字母或大写字母组成。（）

2）若将第8行的“i = 1改为“i = 0”，程序运行时会发生错误。（）

3）若将第8行的“i <= n”改为“i * i <= n”，程序运行结果不会改变。（）

4）若输入的字符串全部由大写字母组成，那么输出的字符串就跟输入的字符串一样。（）

•选择题（2分一个)

5）若输入的字符串长度为18,那么输入的字符串跟输出的字符串相比，至多有（）个字符不同。

6）若输入的字符串长度为（），那么输入的字符串跟输出的字符串相比，至多有36个字符不同。

1.

A. 正确

B. 错误

2.

A. 正确

B. 错误

3.

A. 正确

B. 错误

4.

A. 正确

B. 错误

5.

A. 18

B. 6

C. 10

D. 1

6.

A. 36

B. 100000

C. 1

D. 128

17.填空题（13.5分）

```
#include <iostream>
using namespace std;

long long n, ans;
int k, len;
long long d[1000000];

int main() {
```

```
9    cin >> n >> k;
10   d[0] = 0;
11   len= 1;
12   ans = 0;
13   for (long long i = 0; i <n; ++i) {
14      ++d[0];
15      for (int j = 0; j + 1<len; ++j) {
16       if (d[j] == k) {
17          d[j] = 0;
18          d[j + 1] += 1;
19          ++ans;
20       }
21      }
22      if (d[len - 1] == k) {
23       d[len - 1] = 0;
24       d[len] =1;
25       ++len;
26       ++ans;
27      }
28   }
29   cout << ans << endl;
30   return 0;
31  }
```

假设输入的 n是不超过 2^{63} 的正整数，k都是不超过 10000 的正整数，完成下面的判断题和单选题：

•判断题

1）若 k=1，则输出 ans时，len=n。（ ）

2）若 k>1，则输出 ans 时，len 一定小于 n。（ ）

3）若 k>1，则输出 ans 时，k^len一定大于n。（ ）

•单选题

4）若输入的 n 等于： 10^{15}，输入的 k为 1，则输出等于（ ）。

5）若输入的 n等于205,891,132,094,649（即 3^{30}），输入的 k为 3，则输出等于（ ）。

6）若输入的 n 等于 100,010,002,000,090，输入的 k 为 10，则输出等于（ ）。

1.

A. 正确

B. 错误

2.

A. 正确

B. 错误

3.

A. 正确

B. 错误

4.

A. 1

B. $(10^{30}-10^{15})/2$

C. $(10^{30}+10^{15})/2$

D. 10^{15}

5.

A. 3^{30}

B. $(3^{30}-1)/2$

C. $3^{30}-1$

D. $(3^{30}+1)/2$

6.

A. 11,112,222,444,543

B. 11,122,222,444,453

C. 11,122,222,444,543

D. 11,112,222,444,453

18.填空题（10.5分）

```
1   #include <stdio.h>
2
3   int n;
4   int a[1000];
```

```
5
6   int f(int x)
7   {
8       int ret = 0;
9       for (; x; x &= x - 1) ret++;
10      return ret;
11  }
12
13  int g(int x)
14  {
15      return x & -x;
16  }
17
18  int main()
19  {
20      scanf("%d", &n);
21      for (int i = 0; i < n; i++) scanf("%d", &a[i]);
22      for (int i = 0; i < n; i++)
23          printf("%d ", f(a[i]) + g(a[i]));
24      printf("\n");
25      return 0;
26  }
27
```

•判断题

1）输入的n等于1001时，程序不会发生下标越界。（）

2）输入的 a[i]必须全为正整数，否则程序将陷入死循环。（）

3）当输入为”5 2 11 9 16 10”时，输出为”3 4 3 17 5”。（）

4）当输入为”1 511998”时，输出为”18” 。（）

5）将源代码中g函数的定义(13-16行)移到main函数的后面，程序可以正常编译运行。（）

•单选题

6）当输入为”2 -65536 2147483647”时，输出为（）

1.

A.正确

B.错误

2.

A.正确

B.错误

3.

A.正确

B.错误

4.

A.正确

B.错误

5.

A.正确

B.错误

6.

A."65532 33"

B."65552 32"

C."65535 34"

D."65554 33"

19.填空题（15分）

（最小区间覆盖）给出 n 个区间，第 ii个区间的左右端点是 $[a_i,b_i]$。现在要在这些区间中选出若干个，使得区间 [0, m] 被所选区间的并覆盖（即每一个 $0 \le i \le m$ 都在某个所选的区间中）。保证答案存在，求所选区间个数的最小值。

输入第一行包含两个整数 n 和 m（$1 \le \mathbf{n} \le 5000, 1 \le \mathbf{m} \le 10^9$） 接下来 n 行，每行两个整数 a_i，b_i ($0 \le \mathbf{a_i}, \mathbf{b_i} \le \mathbf{m}$）。

提示：使用贪心法解决这个问题。先用 $O(n^2)$的时间复杂度排序，然后贪心选择这些区间。

试补全程序。

```
1   #include <iostream>
2
3   using namespace std;
```

```
4
5   const int MAXN = 5000;
6   int n, m;
7   struct segment { int a, b; } A[MAXN];
8
9   void sort() // 排序
10  {
11   for (int i = 0; i < n; i++)
12   for (int j = 1; j < n; j++)
13   if (①)
14       {
15          segment t = A[j];
16          ②
17       }
18  }
19
20  int main()
21  {
22   cin >> n >> m;
23   for (int i = 0; i < n; i++)
24      cin >> A[i].a >> A[i]·b;
25   sort();
26   int p = 1;
27   for (int i = 1; i < n; i++)
28      if (③)
29       A[p++] = A[i];
30   n = p;
31   int ans =0, r = 0;
32   int q = 0;
```

```
33   while (r < m)
34   {
35      while (④)
36       q++;
37      ⑤;
38      ans++;
39   }
40   cout << ans << endl;
41   return 0;
42  }
```

1)①处应填（ ）

2)②处应填（ ）

3)③处应填（ ）

4)④处应填（ ）

5)⑤处应填（ ）

1.

A. A[j].b>A[j-1].b

B. A[j].a<A[j-1].a

C. A[j].a>A[j-1].a

D. A[j].b<A[j-1].b

2.

A. A[j+1]=A[j];A[j]=t;

B. A[j-1]=A[j];A[j]=t;

C. A[j]=A[j+1];A[j+1]=t;

D. A[j]=A[j-1];A[j-1]=t;

3.

A. A[i].b>A[p-1].b

B. A[i].b<A[i-1].b

C. A[i].b>A[i-1].b

D. A[i].b<A[p-1].b

4.

A. q+1<n&&A[q+1].a<=r

B. q+1<n&&A[q+1].b<=r

C. q<n&&A[q].a<=r

D. q<n&&A[q].b<=r

5.

A. r=max(r,A[q+1].b)

B. r=max(r,A[q].b)

C. r=max(r,A[q+1].a)

D. q++

20.填空题（15分）

（计数排序）计数排序是一个广泛使用的排序方法。下面的程序使用双关键字计数排序，将n对10000以内的整数，从小到大排序。

例如有三对整数(3,4)、(2,4)、(3,3),那么排序之后应该是(2,4)、(3,3)、(3,4) 。

输入第一行为n,接下来n行，第i行有两个数a[i]和b[i],分别表示第i对整数的第一关键字和第二关键字。

从小到大排序后输出。

数据范围 $1<n<10^7$, $1<a[i],b[i]<10^4$。提示：应先对第二关键字排序，再对第一关键字排序。数组ord[]存储第二关键字排序的结果，数组res[]存储双关键字排序的结果。

试补全程序。

```
#include <cstdio>
#include <cstring>
using namespace std;
const int maxn = 10000000;
const int maxs = 10000;

int n;
unsigned a[maxn], b[maxn],res[maxn], ord[maxn];
unsigned cnt[maxs + 1];
int main() {
    scanf("%d", &n);
    for (int i = 0; i < n; ++i)
        scanf("%d%d", &a[i], &b[i]);
    memset(cnt, 0, sizeof(cnt));
```

```
15      for (int i = 0; i < n; ++i)
16          ①; // 利用 cnt 数组统计数量
17      for (int i = 0; i < maxs; ++i)
18          cnt[i + 1] += cnt[i];
19      for (int i = 0; i < n; ++i)
20          ②; // 记录初步排序结果
21      memset(cnt, 0, sizeof(cnt));
22      for (int i = 0; i < n; ++i)
23          ③; // 利用 cnt 数组统计数量
24      for (int i = 0; i < maxs; ++i)
25          cnt[i + 1] += cnt[i];
26      for (int i = n - 1; i >= 0; --i)
27          ④ // 记录最终排序结果
28      for (int i = 0; i < n; i++)
29          printf("%d %d", ⑤);
30
31      return 0;
32  }
```

①处应填（）

②处应填（）

③处应填（）

④处应填（）

⑤处应填（）

1.

A. ++cnt [i]

B. ++cnt[b[i]]

C. ++cnt[a[i] * maxs + b[i]]

D. ++cnt[a[i]]

2.

A. ord[--cnt[a[i]]] = i

B. ord[--cnt[b[i]]] = a[i]

C. ord[--cnt[a[i]]] = b[i]

D. ord[--cnt[b[i]]] = i

3.

A. ++cnt[b[i]]

B. ++cnt[a[i] * maxs + b[i]]

C. ++cnt[a[i]]

D. ++cnt [i]

4.

A. res[--cnt[a[ord[i]]]] = ord[i]

B. res[--cnt[b[ord[i]]]] = ord[i]

C. res[--cnt[b[i]]] = ord[i]

D. res[--cnt[a[i]]] = ord[i]

5.

A. a[i], b[i]

B. a[res[i]], b[res[i]]

C. a[ord[res[i]]], b[ord[res[i]]]

D. a[res[ord[i]]], b[res[ord[i]]]